UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/693,854	10/24/2003	Rob Relyea	MS1-1780US	3939

22801 7590 10/31/2007
LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER	
WANG, BEN C	
ART UNIT	**PAPER NUMBER**
2192	
MAIL DATE	**DELIVERY MODE**
10/31/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

84

Office Action Summary	Application No.	Applicant(s)	
	10/693,854	RELYEA ET AL.	
	Examiner	Art Unit	
	Ben C. Wang	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *10 August 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,4-17 and 19-34* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 4-17, and 19-34* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated August 10, 2007, responding to the Office action mailed April 10, 2007 provided in the rejection of claims 1-34, wherein claims 1, 17, 23, 26, and 30 are amended, claims 2-3 and 18 are canceled.

Claims 1, 4-17, and 19-34 remain pending in the application and which have been fully considered by the examiner.

It should be noted that status of amended claim 1 is incorrectly stated as "Original". Appropriate correction is required.

Applicant's arguments with respect to claims rejection have been fully considered but are moot in view of the new grounds of rejection – see *Lakshminarayanan* - art made of record, as applied hereto.

Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1, 4-17, and 19-34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Smith et al. (Pub. No. US 2003/0028685 A1) (hereinafter 'Smith') in view of Priya Lakshminarayanan (*The .NET Schema Object Model, December 04, 2002, XML.com - O'Reily Media, Inc.*) (hereinafter 'Lakshminarayanan' - art made of record)

3. **As to claim 1** (Currently Amended), Smith discloses a programming interface embodied on one or more computer readable media having computer-executable instructions for performing steps, comprising:

- generating graphical objects using a first group of services (i.e., P.11, *System.Drawing*; P. 12 – *System.Web.UI*);
- formatting content using a second group of services (i.e., P. 2, *System.Runtim.Sericalization.Formatters*; P. 10, *Document Format Information*;

P. 11, *System.Runtime.Serialization.Formatters*; [0049]; [0065], Lines 7-16; [0077]);

- creating components of the graphical objects using a third group of services (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36).

Smith does not explicitly disclose the followings:

- using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first group of services, the second group of services and the third group of services; and
- integrating the first group of services, the second group of service and the third group of services using a consistent programming model and consistent services across the three service groups.

However, in an analogous art of *The .NET® Schema Object Model*, Lakshminarayanan discloses the followings:

- using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first group of services, the second group of services and the third group of services; and
- integrating the first group of services, the second group of service and the third group of services using a consistent programming model and consistent services

across the three service groups (e.g., P. 1, 1st Par. – This article focuses on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is rich API which allows developers to create, edit, and validate schemas programmatically – on of the few such tools available so far; .2nd Par. – SOM operates on schema documents analogously to the way DOM operates on XML documents. Schema documents are valid XML files that, once loaded into the SOM, convey meaning about the structure and validity of other XML documents which conform to the schema. SOM is indispensable for a certain class of application, like a schema editor, where it needs to construct the schema in memory and check the schema's validity according to the WXS (W3C XML Schema) specifications; 4th Par. – This mapping helps easy use of the API. For a complete listing of all the classes available in the System.Xml.Schema namespace, refer to the .NET® Framework Class Library Reference).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Lakshminarayanan into the Smith's system to further provide the followings:

- using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first group of services, the second group of services and the third group of services; and
- integrating the first group of services, the second group of service and the third group of services using a consistent programming model and consistent services across the three service groups in Smith system.

The motivation is that it would further enhance the Smith's system by taking, advancing and/or incorporating Lakshminarayanan's system which offers significant advantages that on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is a rich API which allows developers to create, edit, and validate schema programmatically – one of the few such tools available so far as once suggested by Lakshminarayanan (e.g., P. 1, 1st Par.).

4. **As to claim 4**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the first group of services, the second group of services and the third group of services share a common event system ([0045] – event handling; [0049], Lines 7-10; [0069]).

5. **As to claim 5**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the first group of services, the second group of services and the third group of services share a common property definition system ([0049], Lines 7-10; [0075]; [0079], Lines 1-10).

6. **As to claim 6**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the first group of services, the second group of services and the third group of services share a common input paradigm ([0092], Lines 6-10; [0088], Lines 4-7; [0093], Lines 3-7).

7. **As to claim 7**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the first group of services, the second group of services and the third group of services share a common system for nesting elements associated with a particular group of services within elements associated with another group of services (Fig. 3; [0052] through [0059).

8. **As to claim 8**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the first group of services includes a service that determines an appearance of the graphical objects ([0030], Lines 4-8 – HTML defines how elements are displayed).

9. **As to claim 9**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the first group of services includes a service that determines a behavior of the graphical objects ([0030], Lines 4-8 – XML is used for defining data element on a Web page).

10. **As to claim 10**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the first group of services includes a service that determines an arrangement of the graphical objects ([0030], Lines 4-8 – HTML defines how elements are displayed).

11. **As to claim 11**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the first group of services includes a plurality of nested elements that define the graphical objects (Fig. 3, element 312 – UI; [0057]).

12. **As to claim 12**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the objects are comprised of one or more elements defined by vector graphical graphics ([0062] – vector graphics functionality).

13. **As to claim 13** (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface and an application program interface wherein the first group of services can define window properties in a markup language without launching a new window ([0061] – A windows forms namespace ("*System.Windows.Forms*") containing classes for creating Windows®-based client applications).

14. **As to claim 14**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the first group of services generates a user interface containing a plurality of graphical objects (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36).

15. **As to claim 15**, (incorporating the rejection in claim 1) (Original), Smith discloses a programming interface wherein the second group of services arranges the graphical objects ([0030], Lines 4-8 – HTML defines how elements are displayed).

16. **As to claim 16**, (incorporating the rejection in claim 1) (Original), Smith discloses a software architecture comprising the programming interface (Fig. 2; [0022]; [0044], Lines 1-30).

17. **As to claim 17** (Currently Amended), Smith discloses an application program interface embodied on one or more computer readable media having computer-executable instructions for performing steps, comprising:

- generating graphical objects using a first group of services (i.e., P.11, *System.Drawing*; P. 12 – *System.Web.UI*);
- formatting content using a second group of services (i.e., P. 2, *System.Runtim.Sericalization.Formatters*; P. 10, *Document Format Information*; P. 11, *System.Runtime.Serialization.Formatters*; [0049]; [0065], Lines 7-16; [0077]);
- creating components of the graphical objects using a third group of services (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-

Col., Lines 5-8, 34-36), wherein the first group of services, the second group of services and the third group of services are integrated via: sharing a common programming model (Fig. 3; [0023]; [0052], Lines 1-6).

Smith does not explicitly disclose using a common markup language across the three services to map classes and properties specified in the markup language to an instantiated tree of objects.

However, in an analogous art of *The .NET® Schema Object Model*, Lakshminarayanan discloses using a common markup language across the three services to map classes and properties specified in the markup language to an instantiated tree of objects (e.g., P. 1, 1st Par. – This article focuses on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is rich API which allows developers to create, edit, and validate schemas programmatically – on of the few such tools available so far; .2nd Par. – SOM operates on schema documents analogously to the way DOM operates on XML documents. Schema documents are valid XML files that, once loaded into the SOM, convey meaning about the structure and validity of other XML documents which conform to the schema. SOM is indispensable for a certain class of application, like a schema editor, where it needs to construct the schema in memory and check the schema's validity according to the WXS (W3C XML Schema) specifications; 4th Par. – This mapping helps easy use of the API. For a complete listing of all the classes available in the System.Xml.Schema namespace, refer to the .NET® Framework Class Library Reference).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Lakshminarayanan into the Smith's system to further provide using a common markup language across the three services to map classes and properties specified in the markup language to an instantiated tree of objects in Smith system.

The motivation is that it would further enhance the Smith's system by taking, advancing and/or incorporating Lakshminarayanan's system which offers significant advantages that on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is a rich API which allows developers to create, edit, and validate schema programmatically – one of the few such tools available so far as once suggested by Lakshminarayanan (e.g., P. 1, 1st Par.).

18. **As to claim 19**, (incorporating the rejection in claim 17) (Original), Smith discloses an application program interface wherein the third group of services includes services to generate geometric shapes ([0048] – the client applications namespace pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional, imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on).

19. **As to claim 20**, (incorporating the rejection in claim 17) (Original), Smith discloses an application program interface wherein the second group of services includes arranging a plurality of data elements ([0030], Lines 4-8 – XML is used for defining data element on a Web page).

20. **As to claim 21**, (incorporating the rejection in claim 17) (Original), Smith discloses an application program interface wherein the first group of services includes: a service that determines an appearance of a graphical object ([0030], Lines 4-8 – HTML defines how elements are displayed); and a service that determines a behavior of the graphical object ([0030], Lines 4-8 – XML is used for defining data element on a Web page).

21. **As to claim 22**, (incorporating the rejection in claim 17) (Original), please refer to claim **13** as set forth above accordingly.

22. **As to claim 23**, Smith discloses a computer system including one or more microprocessors (Fig. 4, element 404 – Processing Unit; [0085], Lines 3-5) and one more software programs (Fig. 4, elements 428 - Application Programs, 430 – Program Modules, 432 – Program Data; [0091], Lines 4-5), the one or more software programs utilizing a interface (Fig. 2, element 142 – Application Program Interface; [0039], Lines 1-4) to request services from an operating system (Fig. 2, element 146(1) – Operating System), the services or programming interface including separate commands to request consisting of the following groups of services:

- a first group of services for generating graphical objects (i.e., P.11, *System.Drawing*; P. 12 – *System.Web.UI*); and

- a second group of services for creating components of the graphical objects (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36), wherein the first group of services and the second group of services are integrated by sharing a common programming model (e.g., Fig. 3; [0023]; [0052], Lines 1-6).

Smith does not explicitly disclose consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first and second group of services.

However, in an analogous art of *The .NET® Schema Object Model*, Lakshminarayanan discloses consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first and second group of services (e.g., P. 1, 1st Par. – This article focuses on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is rich API which allows developers to create, edit, and validate schemas programmatically – on of the few such tools available so far; .2nd Par. – SOM operates on schema documents analogously to the way DOM operates on XML documents. Schema documents are valid XML files that, once loaded into the SOM, convey meaning about the structure and validity of other XML documents which conform to the schema. SOM is indispensable for a certain class of application, like a schema editor, where it needs to construct the schema in memory and check the schema's validity according to the WXS (W3C XML Schema) specifications; 4th Par. – This mapping helps easy use of the API. For a complete listing of all the classes available in the System.Xml.Schema namespace, refer to the .NET® Framework Class Library Reference).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Lakshminarayanan into the Smith's system to further provide consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first and second group of services in Smith system.

The motivation is that it would further enhance the Smith's system by taking, advancing and/or incorporating Lakshminarayanan's system which offers significant advantages that on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is a rich API which allows developers to create, edit, and validate schema programmatically – one of the few such tools available so far as once suggested by Lakshminarayanan (e.g., P. 1, 1st Par.).

23. **As to claim 24**, (incorporating the rejection in claim 23) (Original), Smith discloses a computer system wherein the first group of services includes: a service for defining an appearance of the graphical objects ([0030], Lines 4-8 – HTML defines how elements are displayed); and a service for defining an arrangement of the graphical objects (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application

namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36).

24. **As to claim 25**, (incorporating the rejection in claim 23) (Original), Smith discloses a computer system wherein the second group of services includes services to generate a plurality of geometric shapes ([0048] – the client applications namespace pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional, imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on).

25. **As to claim 26** (Currently Amended), Smith discloses a method comprising: calling one or more first functions to facilitate generating graphical objects (i.e., P.11, *System.Drawing*; P. 12 – *System.Web.UI*); and

a second group of services for creating components of the graphical objects (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36); calling one or more second functions to facilitate formatting content (i.e., P. 2, *System.Runtim.Sericalization.Formatters*; P. 10, *Document Format Information*; P. 11, *System.Runtime.Serialization.Formatters*; [0049]; [0065], Lines 7-16; [0077]), wherein the first functions and the second functions are integrated by sharing a common programming model (Fig. 3; [0023]; [0052], Lines 1-6).

Smith does not explicitly disclose consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first and second group of services.

However, in an analogous art of *The .NET® Schema Object Model*, Lakshminarayanan discloses consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first and second group of services (e.g., P. 1, 1st Par. – This article focuses on <u>an API in the .NET® platform</u>, the <u>XML Schema Object Model</u> (<u>SOM</u>), SOM is rich API which allows developers <u>to create, edit, and validate schemas programmatically</u> – on of the few such tools available so far; .2nd Par. – SOM operates on <u>schema documents</u> analogously to the way DOM operates on XML documents. <u>Schema documents are valid XML files</u> that, once loaded into the SOM, convey meaning about the structure and validity of other XML documents which conform to the schema. SOM is indispensable for a certain class of application, like a schema editor, where it needs to construct the schema in memory and check the schema's validity according to the WXS (W3C XML Schema) specifications; 4th Par. – <u>This mapping helps easy use of the API</u>. For a complete listing of all the classes available in the System.Xml.Schema namespace, refer to <u>the .NET® Framework Class Library Reference</u>).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Lakshminarayanan into the Smith's system to further provide consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first and second group of services in Smith system.

The motivation is that it would further enhance the Smith's system by taking, advancing and/or incorporating Lakshminarayanan's system which offers significant advantages that on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is a rich API which allows developers to create, edit, and validate schema programmatically – one of the few such tools available so far as once suggested by Lakshminarayanan (e.g., P. 1, 1st Par.).

26. **As to claim 27**, (incorporating the rejection in claim 26) (Original), Smith discloses a method further including calling one or more third functions to facilitate creating components of the graphical objects (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36).

27. **As to claim 28**, (incorporating the rejection in claim 26) (Original), Smith discloses a method further including calling one or more third functions to facilitate generating geometric shapes ([0048] – the client applications namespace pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional, imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on) contained in the graphical objects.

28. **As to claim 29**, (incorporating the rejection in claim 26) (Original), Smith discloses a method wherein the first functions facilitate: defining window properties in a markup language without launching a new window ([0061] – A windows forms namespace ("*System.Windows.Forms*") containing classes for creating Windows®-based client applications); and generating a user interface containing a plurality of graphical objects (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36).

29. **As to claim 30** (Currently Amended), Smith discloses a system comprising: means for exposing a first set of functions that enable generating graphical objects (i.e., P.11, *System.Drawing*; P. 12 – *System.Web.UI*); and

means for exposing a second set of functions that enable creating components of the graphical objects (i.e., Fig. 3, element 202 – Client Application; [0048] – the client application namespace pertains to drawing and client side UI functionality; P. 13, Left-Col., Lines 50-52; P. 14, Left-Col., Lines 17-19, Right-Col., Lines 18-20; P. 15, Left-Col., Lines 20-22, Right-Col., Lines 37-39; P. 16, Left-Col. Lines 24-27, Right-Col., Lines 5-8, 34-36), wherein the components of the graphical objects include a plurality of geometric shapes ([0048] – the client applications namespace pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional, imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on), and wherein the first set of functions and the second set of functions are integrated by sharing a common programming model (Fig. 3; [0023]; [0052], Lines 1-6).

Smith does not explicitly disclose consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first and second group of services.

However, in an analogous art of *The .NET® Schema Object Model*, Lakshminarayanan discloses consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first and second group of services (e.g., P. 1, 1st Par. – This article focuses on <u>an API in the .NET® platform</u>, the <u>XML Schema Object Model</u> (<u>SOM</u>), SOM is rich API which allows developers <u>to create, edit, and validate schemas programmatically</u> – on of the few such tools available so far; .2nd Par. – SOM operates on <u>schema documents</u> analogously to the way DOM operates on XML documents. Schema documents are valid XML files that, once loaded into the SOM, convey meaning about the structure and validity of other XML documents which conform to the schema. SOM is indispensable for a certain class of application, like a schema editor, where it needs to construct the schema in memory and check the schema's validity according to the WXS (W3C XML Schema) specifications; 4th Par. – This mapping helps easy use of the API. For a complete listing of all the classes available in the System.Xml.Schema namespace, refer to the .NET® Framework Class Library Reference).

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Lakshminarayanan into the Smith's system to further provide consistent services and using a common markup language to map classes and properties specified in the markup language to an instantiated tree of objects across the first and second group of services in Smith system.

The motivation is that it would further enhance the Smith's system by taking, advancing and/or incorporating Lakshminarayanan's system which offers significant advantages that on an API in the .NET® platform, the XML Schema Object Model (SOM), SOM is a rich API which allows developers to create, edit, and validate schema programmatically – one of the few such tools available so far as once suggested by Lakshminarayanan (e.g., P. 1, 1st Par.).

30. **As to claim 31**, (incorporating the rejection in claim 30) (Original), Smith discloses a system wherein the second set of functions further enable arrangement of the geometric shapes on a page to be rendered ([0048] – the client applications namespace pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional, imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on).

31. **As to claim 32**, (incorporating the rejection in claim 30) (Original), Smith discloses a system further comprising means for exposing a third set of functions that enable formatting content for display ([0030], Lines 4-8).

32. **As to claim 33**, (incorporating the rejection in claim 30) (Original), Smith discloses a system wherein the first set of functions and the second set of functions utilize a common markup language (Fig. 2, element 204 – Data and XML; Fig. 3, element 204 – Data and XML; [0047], Lines 5-14; [0049], Lines 1-5; [0065], Lines 7-16).

33. **As to claim 34**, (incorporating the rejection in claim 30) (Original), Smith discloses a system wherein the first set of functions and the second set of functions share a common event system ([0045] – event handling; [0049], Lines 7-10; [0069]) and a common property definition system ([0049], Lines 7-10; [0075]; [0079], Lines 1-10).

Conclusion

34. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Hejlsberg et al., *Application Program Interface for Network Software Platform* (Pub. No. US 2003/0172196 A1)
- Smith et al., *Application Program Interface for Network Software Platform* (Pub. No. US 2003/0167356 A1)
- Hejlsberg et al., *Application Program Interface for Network Software Platform* (Pub. No. US 2003/0177282 A1)
- Smith et al., *Application Program Interface for Network Software Platform* (Pat. No. US 7,017,162 B2)
- Desoli et al., *System and Method for Supporting Emulation of a Computer System Through Dynamic Code Caching and Transformation* (Pub. No. US 2003/0101439 A1)

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

[illegible]

TUAN DAM
SUPERVISORY PATENT EXAMINER

BCW BW

October 22, 2007